ADOPTING BUILDING AUTOMATION IN WEBLABS
Analysis Of Requirements And Solutions

Ricardo J. Costa, Gustavo R. Alves
LABORIS / Polytechnic Institute of Porto - School of Engineering (ISEP), Porto, Portugal
rjc@isep.ipp.pt, gca@isep.ipp.pt

Domingos S. Santos
Polytechnic Institute of Porto - School of Engineering (ISEP), Porto, Portugal
dss@isep.ipp.pt

Keywords: Remote Experimentation, Weblab, Domotic System, KNX.

Abstract: Several companies have been developing domotic Stds. for building automation, enabling users to locally and remotely control several home devices, like: lights, power sockets, heating, ventilation, and air conditioning systems, among others. Besides contributing to improve the building comfort, these Stds. may also be adopted for other purposes, namely in weblabs used in sciences and engineering remote experiments. To increase the sense of immersion in weblabs, we identify domotic Stds. as a standard solution for turning on/off the power infrastructure and controlling the light and temperature conditions of the physical space where a specific experiment may run, thus approaching the sense of being in the lab facilities while accessing them through the corresponding weblab interface. After identifying the added value to weblabs in terms of power savings and in the control of the environmental conditions, we used our knowledge and the WWW to conduct an extensive search on domotic Stds., and after analysing the results obtained, we choose the most appropriated one to be implemented in a Weblab. Regarding the adopted Std., a proof-of-concept is also described, enabling the control of an halogen lamp and a power socket, using a specific Web interface.

1 INTRODUCTION

Weblabs are used for educational purposes since the mid 90's. According to Aktan et al. (Aktan et al., 1996), it was maybe in 1996 the first time an undergraduate weblab has been made fully accessible through the Internet. This solution contributed to the appearance of the Remote Experimentation concept, defined as a distance learning area that enables the remote control of real experiments using computers connected to the Internet. This remote access to labs, commonly used to support the practical work required in sciences and engineering courses, brought several advantages to education, like: a) users (teachers and students) may access resources not locally available, providing the accomplishment of experiments without any costs; b) experiments may be shared among institutions; c) equipment may be used more often; d) costs per student may be reduced; among others (Alves et al., 2005).
Due to all those advantages, already described in literature, nowadays there is a growing number of weblabs. Since those same labs require specific resources to enable a remote access, several solutions for harmonizing the software and hardware used for implementing them have already been proposed and described. However, the existence of many different technologies difficults the choice for a standard approach to implement a specific weblab both in terms of hardware and software. Usually, when a specific weblab is required, an immediate and particular technical solution is adopted for this development. Moreover, due to the specificity of each solution, usually only qualified people are able to develop one, which partially justifies that almost all weblabs fall into the engineering domain (Jing Ma and Jeffrey, 2006). Thus, harmonization at hardware and software levels is an important aspect to take into consideration as to facilitate the construction of standard and well defined weblabs. If harmonization is followed, software/hardware modules used in each weblab may be reused on others. This will ease the creation of weblabs and promote its widespread use in the teaching and learning domain. Moreover, by

following a standard architecture, other aspects may be considered during a weblab implementation, namely the environment of the physical space occupied by it and the power infrastructure. By controlling these two common aspects to all weblabs, further control facilities are given to remote users. They will be able to control the place where a specific experiment is running, like if they were in that place, contributing to approach the in-place lab facilities to weblabs. To implement these aspects in any weblab, we propose the adoption of a standard domotic system bus usually implemented in smart houses, which will ease the control of all the environmental conditions encountered in any lab.

The rest of this paper is organized as follows: in the next section we discuss some requirements associated with weblabs, namely issues concerning the control of the physical space (light incidence and temperature) and the power sockets where the lab devices are connected to; section 3 proposes an architectural solution, based on a domotic system bus, addressing the previous identified requirements; section 4 presents some 'de-facto' domotic Stds. candidate to be used in a weblab; and section 5 describes the solution adopted for our lab that enables remotely controlling two specific devices (a halogen lamp and a power socket). The paper ends with some considerations about the work already done and future directions.

2 ANALYSIS OF REQUIREMENTS

Each weblab requires a specific place to accommodate the apparatus, the measurement equipment and the servers. Usually, those places have characteristic light incidence, temperature and humidity conditions, containing all the equipment power supplied. Figure 1 illustrates a possible place to host a remote experiment, surrounded by several equipment also able to control remotely.

The necessity to adopt a weblab to support the practical work, 24 hours per day, 7 days per week, has consequences in the power consumption and in the results obtained from the experiments. Specific attention should be paid to the physical environment where an experiment is running, namely with the light and temperature conditions. Supposing that a weblab facility has direct sun light and visual feedback is required, then artificial lights are not mandatory, i.e. they are not required to be switched on during the day.

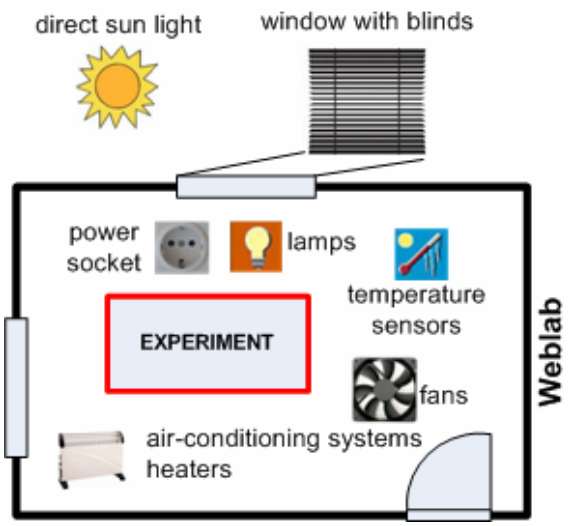

Figure 1: Physical space associated to a weblab.

Eventually, controlling the blinds can provide better light conditions depending on the part of the day and of the present season. During the night and when not in use, the lights at the weblab may also be switched off in order to save energy. Besides, depending on the type of prototype and equipment used in the remote experiment, the ability to control/monitor the temperature may be required. For that purpose, two solutions may be considered: the temperature control can be made automatically within the weblab, or users may be provided with the ability to control it remotely. In the first solution, an air-conditioning system, inside the weblab, controls the temperature and continuously guarantees that pre-specified conditions are satisfied for a correct lab operation. Alternatively, the temperature can be monitored by a sensor and, based on the readings, users may change it for a specific value, by remotely controlling fans, heaters or even specifying a new temperature value for the air-conditioning system. Although not being a common request in a remote experiment, same weblabs may require a precise temperature control. This is the case with experiments that use specific materials that change their properties according to temperature. Another situation, where temperature control may be required, is in some environments where the measurement devices require a given temperature in order to work properly. In both situations, remote users should have specific information about the equipment, so as to adjust the present temperature conditions to the required ones.

An additional requirement that should be considered is the ability to switch off each device of the weblab, when not in use. If possible of being done remotely, the all lab may be switched off when a specific ex-

periment ends. Latter, when another experiment is starting, the remote user may turn it on. This will contribute to a reduction of the power consumption and hence of the energy costs. Additionally, it may also be desirable to reinitialize a certain device by applying an off/on sequence. These suggestions/requirements depend on whether the device and apparatus to be switched off/on requires a setup procedure or not. Considering a PC, for instance, it is not possible to switch it off by just pulling out the plug from the power socket, i.e. by simply switching it off. If in some devices turning off the power will take them into a default state, on others we have to follow a specific sequence for that purpose. This is usually the case with the apparatus, which sometimes requires a software initialization made by an Instrumentation Server. In this situation, if we intend to setup the entire weblab by resetting it, first we have to reset the Instrumentation Server and then the apparatus, so it can be reinitialized by using control signals coming from the Instrumentation Server. Again, specific attention is required in the way the Instrumentation Server is reset, since it can not be turned off abruptly because it could damage the software installed on it. Preferably, the server should be reset by an UPS (Uninterrupted Power Supply) using a soft reset that allows turning off the server by software. If all these setup and reinitialization considerations are supported through remote control, then the technical support, usually made by a technician, may be reduced or even suppressed, which also contributes to reduce the weblab maintenance costs. Moreover, switching off all devices and the apparatus, when not in use, reduces the ageing effects, which also contributes to the quality of the results obtained from the remote experiments.

To address all the previous points, we propose using a specific architecture with a standard domotic system bus, usually implemented in smart houses. The advantage of using a standard solution is that all institutions currently supporting weblabs may quickly adopt it with small development efforts.

3. PROPOSED ARCHITECTURE

Commonly, weblabs are idealized for specific experiments. Although several weblab architectures have been described in related literature, to the best of our knowledge, few have considered the ability to remotely control the physical space as well as the power sockets where the lab devices connect to. At this point two situations are common: 1) this requirement is not even considered; 2) in-house specific solutions are used to tackle the problem. To address both situations we propose weblab designers to considerate the possibility of using a standard domotic system bus, in the way illustrated in figure 2.

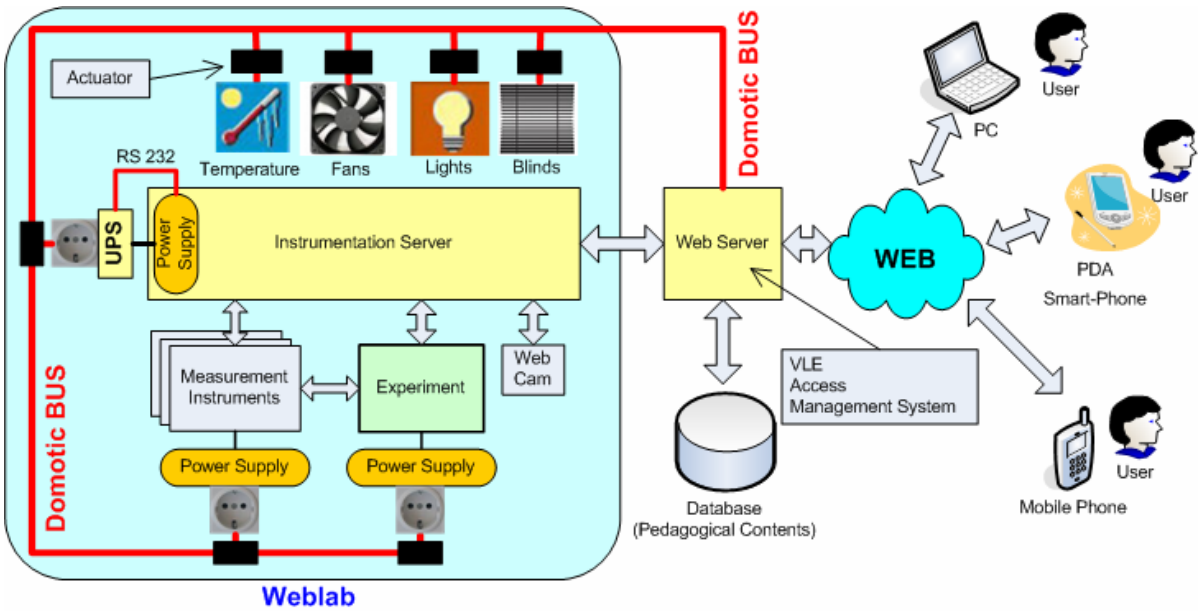

Figure 2: Suggested architecture to control the temperature and light conditions plus the power infrastructure (power sockets) of the weblab physical space.

This proposed architecture contains two servers: the Instrumentation Server and the Web Server, both connected through an Intranet. All devices available in the weblab, namely the experiment apparatus, Webcams, and the measurement equipment are controlled by the Instrumentation Server. In order to facilitate the development of a weblab, all devices have to be autonomous, i.e. they should be able to work in an independent way, which means that all the hard processing tasks should be made inside each one. The Instrumentation Server only sends/receives basic signals to control and monitor each device. At the same time, the Instrumentation Server transfers information to/from the Web Server enabling the interface between remote users and the lab. Another relevant aspect presented in the proposed architecture is the UPS. This unit is connected to the Instrumentation Server via a serial RS232 connection, and to a power socket controlled by the domotic system bus. By turning off this power socket, the UPS will send a signal to the Instrumentation Server, instructing him to initiate a software shut down sequence. This avoids the application under execution in the Instrumentation Server, from being damaged because of a sudden power failure (i.e. an off command) or an improper shut down sequence.

Users must establish a Web connection to the Web Server so they can download the web interfaces designed for a specific device (PC, PDA, Smart Phone or Mobile phone), to control the weblab. The Web Server should also contain the relevant pedagogical contents, as weblabs experiments are mainly used for transferring knowledge and for practicing theoretical concepts (Jing Ma and Jeffrey, 2006). This means that a database is required together with a Virtual Learning Environment both intending to keep e-learning course contents (e.g. documents and images) together with specific tools for course management, like assessment tools and others. Moreover, since usually only one experiment is available in a specific weblab, the Web Server should also have an access management system, required to handle simultaneous accesses (Ferreira and Cardoso, 2005).

Additionally, the Web Server should also control the domotic system bus. For this purpose, it must be coupled to the domotic system bus so that a software layer installed in it may enable remotely controlling all lab devices. This facility should be made by the Web Server because it is the only device required to be permanently turned on, to allow the weblab availability 7 days per week, 24 hours per day. Thus, by adopting this architecture it is possible to remotely turn on/off the entire weblab and control the physical space.

Specific domotic devices named actuators are directly connected to lamps, temperature sensors, and other devices, and are characterized by having a dedicated processor that enables doing specific control tasks required for each of those devices, like: increasing lights intensity, open/close blinds, controlling the temperature and so on. Besides controlling each device, the processor within the actuator has also the ability to understand a communication protocol, specific of the domotic bus. Adopting this solution, the Web Server will be free from the hard processing tasks. This will facilitate the adoption of this architecture for all weblabs, because it is not necessary to develop the hardware, and the software will be much easier to build. Besides, if we adopt a commercial domotic system, which guaranties a well tested solution, user's confidence to use the weblab will increase.

However, the choice for the most appropriated domotic system bus to implement in a weblab is not straightforward because there are several companies developing dedicated protocols to control lights, blinds, temperature, and other devices. So, before adopting a specific domotic system to control the environment and the weblab power infrastructure, a research about the most popular and used domotic Stds. is recommended.

4. THE MAIN DOMOTIC STDs.

Based in a web research, this section presents some of the most well known 'de-facto' standards available in the market for building automation. Our research only concentrates in Stds. that have already some implementation of power automation products which enable controlling: power sockets, lighting controllers, temperature sensors, fans, heaters, air-conditioning systems, etc.

To choose a Std. we identified the medium often used to transmit information among devices (P- powerline; W- wireless; or D- dedicated bus), and classified some characteristics that we believe to be important to decide about which is the most appropriated for serving a weblab implementation, namely:

- Reliability: communication among devices connected to the bus is secure, reliable and without interferences (the communication medium also influence this characteristic, considering that a dedicated bus is more reliable than a power line and a wireless connection);

- Products: the amount of products available in the market (only considering power automation products, like: push buttons, sensors, light controllers, etc..) ;
- Price: costs of products and installation;
- APIs/software: software applications and libraries (APIs) available for system control, which are required to implement the architecture presented in figure 2;
- Support: number of companies and associations supporting the protocol, and if the protocol is a real Std..

All these characteristics are classified in table 1 from 0 (less appropriated) to 5 (more appropriated). For each Std., we made a sum of all classifications in order to facilitate the choice for the most appropriated to implement in a weblab.

NOTE: The readers interested in further considerations about each Std. may consult the web site available in the table.

Table 1: Comparing some 'de-facto' domotic Stds..

Standard / Protocol	Medium	Reliability	Products	Price	APIs/software	Support	Total
UPB (www.pcslighting.com)	P	4	2	4	3	3	16
X-10 (www.x10.com)	P	2	4	4	4	4	18
INSTEON (www.insteon.net)	P (W)*	4	2	4	3	3	16
Z-Wave (www.z-wavealliance.org)	W	4	3	3	4	4	18
Zigbee (www.zigbee.org)	W	4	1	0	4	4	13
LonWorks (www.echelon.com)	D (PW)*	5	3	2	4	4	18
KNX (www.konnex.org)	D (PW)*	5	5	2	4	5	21

*- the letter(s) between parentisis means that although the Std. describes the referred transportation medium, often it is not used for building automation.
(Table constructed upon a web research and the document referred in (Bates et al., 2006)).

All Stds. use different communication mediums to transmit information among devices. The adoption for one solution depends of the available lab infrastructure. However, if possible, we suggest using dedicated buses, since they have lower electromagnetic interferences and are more secure than wireless and powerline communications. Besides the quality of the protocol used by each Std., the *medium* contributes for the classification given for the *reliability*.

At this point, LonWorks and KNX have a higher score classification since both have a reliable and secure protocol and usually use a dedicated bus to transmit information among devices. It is also important to observe the X-10 low classification because, as described in many literature, it does not have a collision avoidance mechanism and may suffer interferences from electronic devices.
At least one API is available to control each protocol. However, the INSTEON and UPB Stds. have a lower classification in the *APIs/software* because we didn't find much software or APIs to control them using a PC. All the other Stds. have at least more than one API or software tool to control the connected devices (e.g DLLs, Java APIs, etc.). Another considered aspect is the hardware kits that may be used to establish the interface between a PC and the bus. We found many kits for the LonWorks, Z-Wave and Zigbee protocols.
The *support* reaches the highest classification in the KNX Std. because it is supported by many companies and it is an European and world Std.. The same happens with X-10, Z-Wave, Zigbee and LonWorks, however none is a real Std.. Since UPB and INSTEON are restricted to the USA market, and are developed by specific companies, we decided to give them a lower classification.
Comparing to the other Stds., KNX is the one with more power home *products*. In this issue, UPB and INSTEON have another low classification since both protocols are only implemented for USA. X-10 has many products for USA and more recently for Europe too, which justifies its high classification. It is also important to refer that we did not find any power home products for the Zigbee Std.. However, literature refers that there are some products and also alert that in the future many products will be available. This situation justifies the lower classification given in *products*.
The *price* is also an important characteristic to take into consideration for a Std. adoption. Here, we may say that reliability is expensive, because KNX and LonWorks have the most expensive products and were classified as more reliable than the others. Besides, since we did not find any products for Zigbee, we have decided to give a zero to the price characteristic.
Totalizing the classifications of each Std., the preferred one to be implemented in a Weblab is the KNX Std.. It has a high reliability, because the majority of products adopt a dedicated bus to transmit information between devices and the protocol uses a CSMA/CD collision detection. Besides, KNX is an open Std. for home and building control, being both

an European Std. (CENELEC/EN 50090 and CEN/EN 13321-1) and a world Std. (ISO/IEC 14543-3), it is platform independent, guaranties interoperability of products from different manufacturers and stands for high product quality guaranteed by the KNX Association (KNX, 2007). KNX/EIB denomination is often used in literature because the KNX standard is based in the communication stack of the EIB specification, and it also integrates new configuration mechanisms and communication media originally developed by other domotic specifications, namely Batibus and EHS. The only disadvantage of KNX is the price of products, which are usually higher when comparing to other devices used by the domotic Stds.. However, based on the presented analysis, we have decided to implement a solution with KNX at our lab, which specifically enables remotely controlling a halogen lamp and the power supply that powers the experiment apparatus.

5. IMPLEMENTED LAYER

In our implementation, we adopted the architecture presented in figure 2 (section 3) using KNX devices, and we installed in the Web Server a software layer to control the KNX system. Instead of implementing the KNX protocol from the scratch, we developed the software layer using a JAVA API named CALIMERO developed by the Vienna University of Technology, Automation Systems Group (Calimero, 2007). This API provides a set of methods to control the KNX bus communications and all the connected devices. The next sub sections present all the development aspects at both the hardware and software levels.

5.1 Physical Devices and Their Configuration

A simple domotic system bus was implemented as illustrated in figure 3, which presents the set of KNX devices used in our lab that, locally and remotely, allows to: 1) turn on/off and adjust the brightness of a halogen lamp; 2) switch on/off a power socket. Figure 4 contains some photos of those devices.

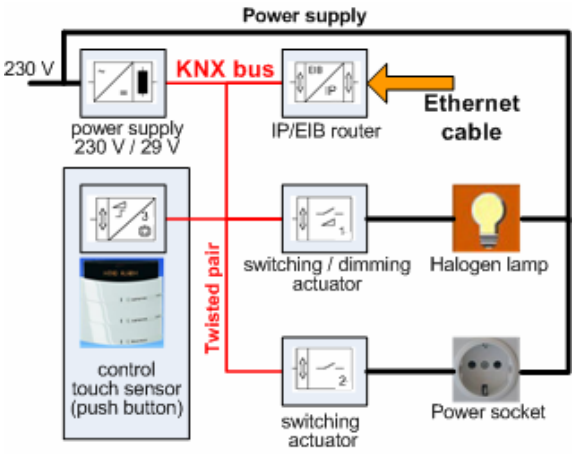

Figure 3: Domotic system implemented in the weblab.

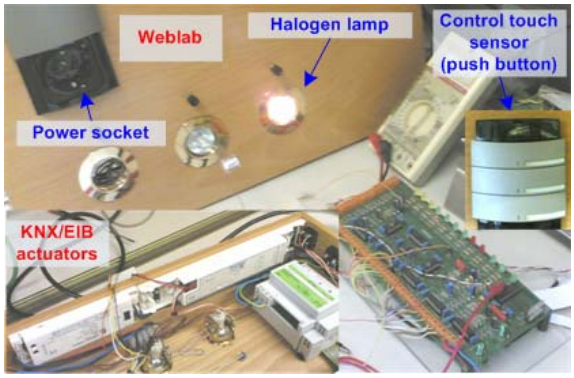

Figure 4: Pictures of the KNX domotic system used in the weblab.

Besides the power supply required for the KNX bus, this topology contains two device types: two actuators (dimming and switching actuators) and a 3 gang push button (control touch sensor). The actuators are directly connected to the elements that will be remotely/locally controlled (the halogen lamp and the power socket) using the push button. By using each button of this sensor, users may locally turn on/off the power socket, and the halogen lamp and control its brightness, as illustrated in figure 5.

We connected the KNX bus to the Web Server using an IP/EIB router. This router converts all the IP frames, sent by the Web Server, to EIB frames, to control and monitor each element. Table II specifies the actual devices used in our lab, the manufacturer, their functions, and the actual market price (relative to May 2007). Notice that others may be chosen, with the same implemented functionalities and eventually with lower prices.

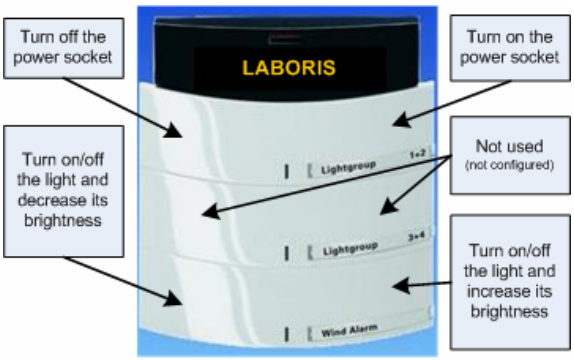

Figure 5: Push button used to locally control the devices connected to the domotic system bus.

Table 2: Domotic devices used in the weblab

Device	Manufacturer	Functionality	Price(€) (May 07)
Tronic dim actuator 1gang 215W built-in (75331002)	Berker	Controls the brightness of the halogen lamp and allows turning it on/off	310.80
Switch actuator 2gang 10A built-in (75332001)	Berker	Turns on/off the power socket	271.43
IP/EIB router (AP 146 /5WG1 146-3_B01)	Siemens	Converts IP to EIB frames and interconnects the KNX bus to an Intranet/Internet	769.23
Power supply (5WG1 122-1AB01)	Siemens	Supplies the KNX domotic system bus	314.63
Push button (6323 3f-triton-switch sensor)	ABB	Allows controlling locally the halogen lamp and the power socket	182.00
			∑ = 1848.09

All the devices available in the domotic system bus were previously configured by a software package named ETS (ETS, 2007). With this software installed in a common PC with the Windows XP OS, we: a) defined an address for each KNX device so they can be recognized in the bus; b) specified the operation of each device, configuring its parameters (e.g. specified that a simple touch in one button of the sensor will turn on/off the lamp); and c) implemented a logic connection between the devices to allow their communication.
After configuring the bus and the devices, we had to download that configuration to the system by connecting, through a local network, the ETS to the KNX bus. This transference required the use of the IP/EIB router using a specific IP and port address. Figure 6 presents a screenshot with the configurations made in the ETS system to program each KNX device.

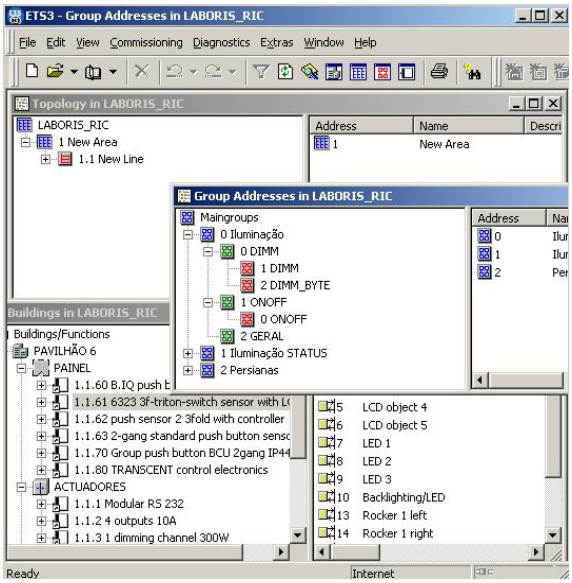

Figure 6: Configuration of the domotic system using the ETS software.

The presented solution contains the usual steps made when a simple KNX system is configured to be used locally. For the remote control, another software layer was specified to control all KNX devices (as locally done through the push button).

5.2 Remote Control

There are in the market some IP gateways that allow controlling the KNX system through the Internet. However, all have in common rigid Web interfaces which are not able to change. This means that if we want to build a specific interface, we have to face with several buttons eventually not required for a specific experiment. Besides, using a gateway will be expensive (its price rounds 1000 €). Then, if we use the required Web Server to remotely control the KNX bus, the solution will be cheaper and will facilitate the construction of appropriated Web interfaces for each remote experiment.
Thus, we developed a new software layer to control the lamp and the power socket available in the weblab. Besides controlling the KNX system, the architecture is modular and may be used by any experiment, since it works together with any existing we-

blab. Figure 7 presents the software modules implemented at our lab, where a particular attention should be paid to the Web Server and to the domotic bus connections.

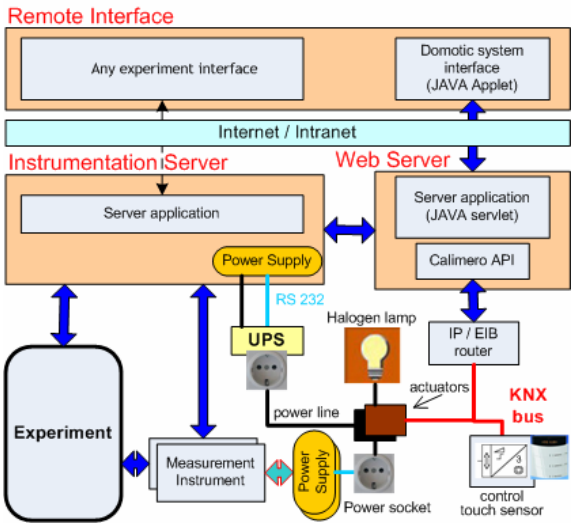

Figure 7: Implemented software architecture to control the KNX/EIB devices.

The architecture uses two servers with specific software, both following a client-server approach.
Since the Instrumentation Server is attached to the weblab apparatus, it should have a software application to control the lab. However, if we want to remotely control the lab, the Instrumentation Server must also have an HTTP (HyperText Transfer Protocol) server installed, so users may download specific Web interfaces. However, an HTTP server is not always required to be installed in the Instrumentation Server. It may have a resident application communicating with another application running in the Web Server. In this situation, all the Web interfaces will be accessed through the Web Server where an HTTP server is required. With this solution, instead of having two open Web accesses, one to control the domotic system bus and the other to control the experiment, we only need one to control both. All commands received in the Web Server, used to control the experiments, are redirected to the application installed in the Instrumentation Server. The disadvantage of this second solution is the requirement to develop two distinct applications to control the lab. However, we will only need one open Web connection, which promotes higher security.
We developed a client-server application for the Web Server that enables remotely controlling the KNX domotic system bus. As previously said, this application is required to be in the Web Server because it controls the power socket of the UPS connected to the Instrumentation Server, i.e. if the software application was installed in the Instrumentation Server the user would not be able to turn it on/off neither to reset it. To avoid developing all the logic behind the KNX/EIB protocol, we used the JAVA API Calimero integrated in a JAVA server application (Servlet). However, to use the Calimero API, an IP/EIB router was required to transmit information from the Servlet application to the KNX/EIB bus. All details behind the KNX protocol are hidden in this API together with the router, facilitating development tasks. Besides, we didn't need to place the router near the Web Server, since all data control generated by the Servlet are transported through an IP network. The only requirement during the development was the use of the specific port 3671 to access the IP/EIB router. In other words, we can not use a firewall that blocks port 3671 between the router and the Web Server, in particular with the Servlet application. Since this application was developed using JAVA software and as it follows a client-server architecture, the Web Server runs an HTTP server named TomCat (TomCat, 2007) to understand and process user's requests together with the Java Servlet Technology (Servlet, 2007). When a specific user wants to access the lab, he/she makes an HTTP access to the Web Server using a particular IP address, and a Java Applet, illustrated in figure 8, will be downloaded to his/her PC.

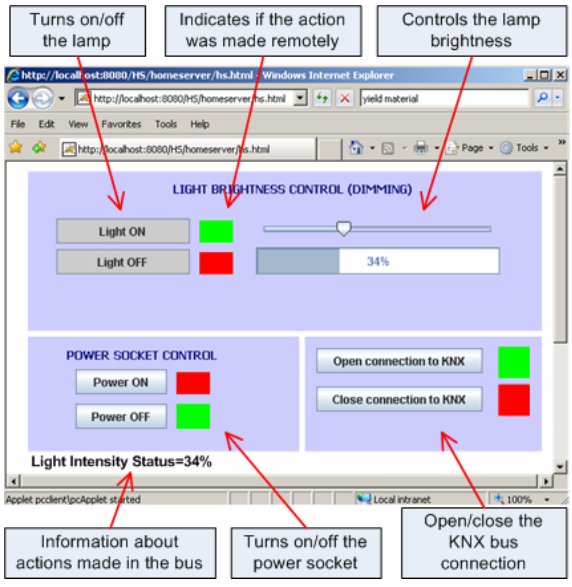

Figure 8: Web interface (Java Applet) used to control the KNX domotic system bus.

Any interface developed to control a specific experiment may work together with this interface that enables users to remotely control the domotic system bus. To use this last interface, an user must establish a connection to the KNX/EIB bus, by pressing the buttons available in bottom-right corner (fig. 8). After connected to the bus, a message notifying the users will appear at the bottom of the interface, indicating a successful connection or any problem that may have occurred. Assuming a successful connection, remote users are able to control the KNX devices (lamp and power socket) using the buttons to turn them on/off and the slide bar to regulate the lamp brightness. Since we adopted the HTTP as the communication protocol between the Servlet application installed in the Web Server and the Java Applet, any change made locally may be actualized in the Applet by making a refresh. This refresh procedure, which may be done by simple reloading the Applet, is made automatically every 30 seconds, updating the interface for user's monitorization.

All interactions made between users and the Web Server are described in figure 9, illustrating the step sequences from initially connecting to the Web Server until finally controlling the KNX devices:

1. Using a Web browser, users access a Web page located in the Web Server.
2. Within this page, a Web interface (an Applet) is downloaded to the user's PC.
3. Once the Web interface has been downloaded and opened, users have the ability to control the lamp brightness and the power socket of the weblab by sending commands to the Servlet application.
4. The Servlet sends those commands to the IP/EIB router, in order to control the devices.
5. For a local control, technicians may use the push button that works independently of the server application.

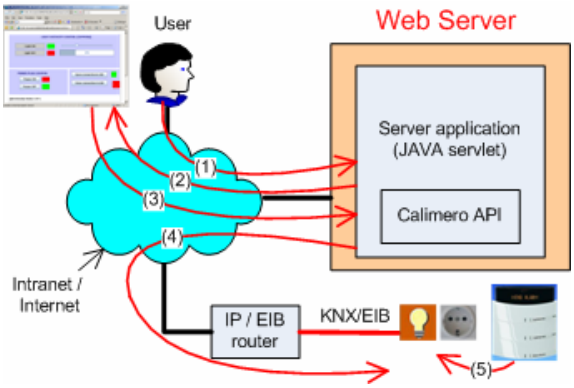

Figure 9: Process sequence used to control, locally and remotely, the KNX devices.

6. CONCLUSIONS AND FUTURE WORK

Educational requirements promoted the development of several software tools to support the teaching and learning processes. These tools, running on PCs connected to the Internet, facilitated the remote control of equipments promoting the emergence of the Remote Experimentation concept, which allows students/teachers to access real labs in order to run experiments, almost in the same way as they would do locally. This solution brought many benefits to education providing versatility, since anyone may access a real lab at anytime from everywhere, while at the same time reducing the costs in education.

However, to provide a weblab it is necessary to implement a hardware/software infrastructure. Usually, these implementations do not follow a standard solution (hardware and software) and the control of the physical space together with the power sockets are often forgotten. If these remote control facilities are implemented in a weblab, the energy costs may be reduced (the apparatus are able to be turned off when not in use) and the quality of the experiments may also increase, since all the physical space (light, temperature, and others) may be controlled and monitored. To provide all those facilities in a weblab, this paper proposed the use of a standard domotic system. Since there are many Stds. available in the market, it was necessary to conduct a research about the most appropriated one. Based on that research, and subsequent analysis, the KNX was selected as the most appropriated domotic system Std. for using in a weblab. By adopting this system, it will be easier to implement a uniform weblab architecture and it will be straightforward to implement additional control facilities to the remote users. Based on all these advantages, this paper presented an implementation of the KNX system bus in a weblab, enabling to control through the Internet a halogen lamp and a power socket using a JAVA Web interface (an Applet). By developing specific Web interfaces, this same approach may be adapted to other users' devices, like PDAs, SmartPhones or Mobile Phones, increasing the access versatility, by enabling students/teachers to use recent mobile devices. A log database, keeping all the actions made in the KNX devices, may also be implemented in the future, bringing benefits to assessment, since it will facilitate teachers to understand some of the actions made by students in the weblab.

To conclude, this paper intended to be an alert to the ability of controlling other aspects usually forgotten in Remote Experimentation, namely the physical

environment and the power sockets used to supply a weblab. Moreover, adopting the reliable and well implemented KNX domotic system bus, will promote the construction of a uniform architecture using commercial devices. However, it is important to consider some recent Stds. that tends to proliferate in the near future, by the appearance of new home automation products.

ACKNOWLEDGEMENTS

This work was only possible with the collaboration of Intelbus Lda. (Intelbus, 2007) that provided some of the KNX devices used in the presented solution.

REFERENCES

B. Aktan, C. A. Bohus, L. A. Crowl and M. H. Shor, 1996. *Distance Learning Applied to Control Engineering Laboratories.* In IEEE Transactions on education, vol.39, nº 3.

G. Alves, J. M. Ferreira, D. Müller, Heinz-H Erbe, N. Hine, J. B. M. Alves, C. Pereira, L. Chiang, O. Herrera and E. Sucar, 19 - 22 Sept., 2005. *Remote Experimentation Network - Yielding an Inter-University Peer-to-Peer e-Service.* In 10th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA'05), Catania, Italy.

Jing Ma and Jeffrey Nickerson, 2006. *Hands-on, Simulated, and Remote Laboratories: A comparative Literature Review.* In ACM Computing Surveys, vol. 38, nº 3.

J. M. Ferreira and A. Cardoso, 2005. *A Moodle extension to book online labs.* In International Journal of Online Engineering (iJOE), vol. 1, nº. 2.

Bates, R., Daunys, G., Villanueva, A., Castellina, E., Hong, G., Istance, H., Gale, A., Lauruska, V. Spakov, O., and Majaranta, P., 2006. *D2.4 A Survey of Existing 'de-facto' Standards and Systems of Environmental Control.* In Communication by Gaze Interaction (COGAIN), IST-2003-511598: Deliverable 2.4, available at www.cogain.org/results/reports/COGAIN-D2.4.pdf.

KNX, 2007. *Association that promotes the KNX standard for Home and Building Control.* Available at: www.konnex.org.

Calimero API, 2007. *EIBnet/IP is a client library that supports tunneling connections to the KNX/EIB bus. A*vailable at: calimero.sourceforge.net.

ETS, 2007. *Tool Software design to configure intelligent home and building control installations made with the KNX system.* Available at: www.konnex.org/knx-tools/ets/intro.

TOMCAT, 2007. *Apache Tomcat Web Server.* Available at: tomcat.apache.org.

Java Servlet Technology, 2007. *Java Servlet technology provides Web developers a consistent mechanism for extending the functionality of a Web server and for accessing existing business systems.* Available at: java.sun.com/products/servlet.

Intelbus Lda., 2007. *Company that implements solutions in the domotic area.* Available at: www.intelbus.pt.